Istruzioni Macchina - Pila

Corso di Architettura degli elaboratori e laboratorio

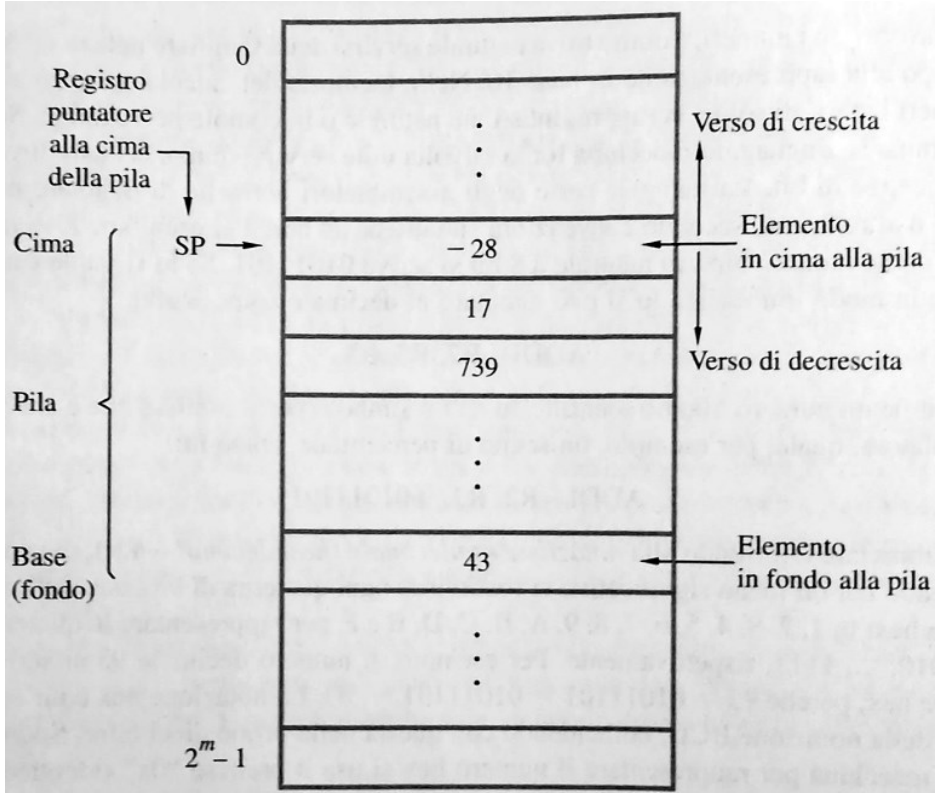
Modulo Laboratorio

Gabriella Verga

Gestione di pila

Pila (Stack)

- La pila consiste in una lista di elementi, alla quale però si possono applicare alcune restrizioni circa il modo di inserirli ed estrarli.
- L'estremità in alto è detto cima (o top), l'estremità in basso base o fondo (bottom).
- **Si può lavorare solo agendo sulla cima della pila.**
- Stack Pointer (SP): registro che punta alla cima della pila [R13].



Operazioni

E’ possibile eseguire solo due operazioni:

1. **PUSH** (o impila): porre un elemento in cima

   Subtract SP, SP,#4 ;diminuire l’indirizzo contenuto in SP di una parola per puntare alla nuova cima

   Store Rj, (SP) ; scrivere il valore richiesto nella parola puntata da SP

2. **POP** (o spila): togliere un elemento dalla cima

   Load Rj, (SP) ;copiare il valore contenuto nella locazione di memoria puntata da SP in un registro del processore

   Add SP, SP,#4 ;aumentare l’indirizzo contenuto in SP di una parola per puntare alla nuova cima

Operazioni in Visual

Istruzione	Operazione	Note
STM[codice] SP[!] {lista di registri}	PUSH	Store Multiple Registro di ordine più basso in cima
LDM [codice] SP[!] {lista di registri}	POP	Load Multiple Registro di ordine più basso dalla cima

Codice	Descrizione
FD	Full descending
FA	Full ascending
ED	Empty descending
EA	Empty ascending
!	Aggiorna il valore del puntatore alla cima a fine operazione

STM

Supposto che R0 contiene 18, R1 contiene 20, R2, contiene 22, R3 contiene 24, R4 contiene 26, R5 contiene 28. Effettuiamo delle operazioni di Store Multiple

FD {R0,R1}
FD {R2,R3}
FD {R4,R5}

SP →	26
	28
SP →	22
	24
SP →	18
	20

FD {R0,R1}
FD {R3,R2}
FD {R4,R5}

SP →	26
	28
SP →	22
	24
SP →	18
	20

FA {R0,R1}
FA {R2,R3}
FA {R4,R5}

	18
SP →	20
	22
SP →	24
	26
SP →	28

ED {R0,R1}
ED {R2,R3}
ED {R4,R5}

SP →	
	26
SP →	28
	22
SP →	24
	18
	20

EA {R0,R1}
EA {R2,R3}
EA {R4,R5}

	18
	20
SP →	22
	24
SP →	26
	28
SP →	

LDM

Supposto che R0 contiene 18, R1 contiene 20, R2, contiene 22, R3 contiene 24, R4 contiene 26, R5 contiene 28. Effettuiamo delle operazioni di Load Multiple

SP →	26
	28
	22
	24
	18
	20

R0	0
R1	0
R2	0
R3	0
R4	0
R5	0

LDM {R0,R1} = LDM {R1,R0}

	26
	28
SP →	22
	24
	18
	20

R0	26
R1	28
R2	0
R3	0
R4	0
R5	0

LDM {R3,R4}

	26
	28
	22
	24
SP →	18
	20

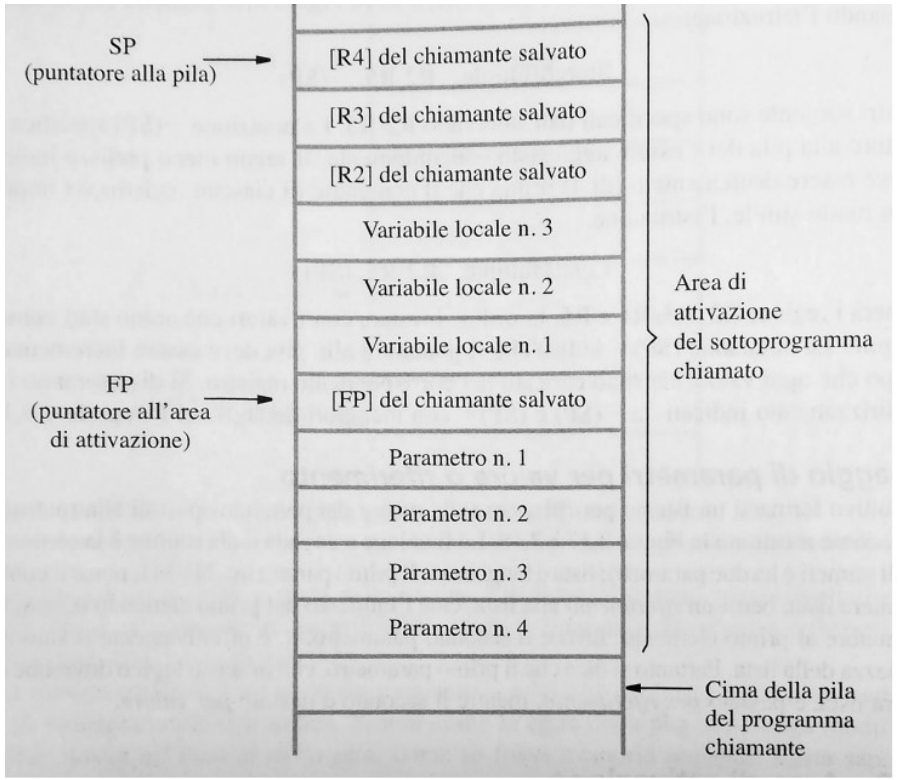
R0	26
R1	28
R2	0
R3	22
R4	24
R5	0

Esempio

1. Eseguire l'operazione 20 + 10 - 15 e salvare il risultato in memoria, usando la pila e conservando i vecchi valori contenuti nei registri.

Area di attivazione in pila

- Il blocco di memoria nella pila riservato al sottoprogramma è chiamato Area di attivazione (**Stack Frame**)
- Il **Frame Pointer** FP è un registro che punta allo Stack Frame del sottoprogramma in esecuzione.
- *Lo Stack Frame contiene: i parametri, il FP del programma chiamante, le variabili locali e valori di registri salvati*



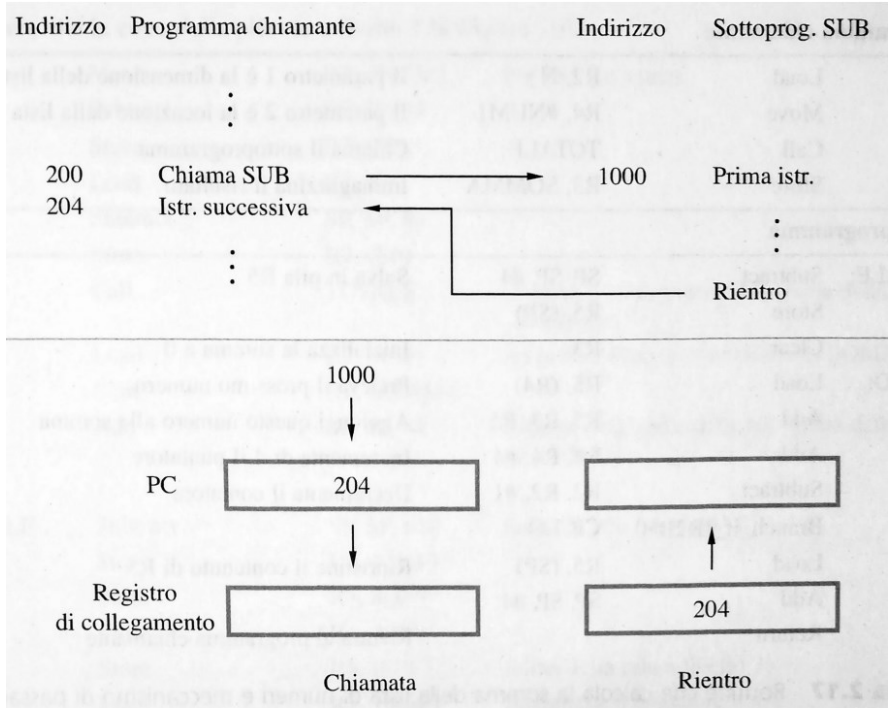
Gestione di sottoprogramma

Sottoprogramma

- Un **sottoprogramma** o routine è una lista di istruzioni che eseguono un compito specifico e che possono essere richiamate in un qualsiasi momento durante l'esecuzione di un programma.
- Il programma che effettua il salto è detto **chiamante** (called program o caller) mentre il sottoprogramma a cui si passa a sottoprogramma è speciale ed è detta istruzione di chiamata a sottoprogramma (call instruction) o a **routine**.
- Terminata la routine è necessario riprendere l'esecuzione dall'istruzione successiva, si dice che la routine rientra al programma chiamante e l'istruzione macchina che ha effettuato questo passaggio cruciale è detto istruzione di rientro.
- Il **Link Register** (LR) è un registro speciale in cui si **memorizza l'indirizzo dell'istruzione di rientro** durante l'esecuzione di un sottoprogramma

Sintassi

- **Chiamata a sottoprogramma:**
  - **Call INDIRIZZO**
- L'operazione di chiamata esegue due passi:
  1. *Salva il contenuto del registro PC nel Link Register;*
  2. *Salta all'indirizzo di destinazione indicato nell'istruzione di chiamata.*
- **Istruzione di rientro:**
  - **Return**
- L'istruzione di rientro salta all'indirizzo di rientro contenuto nel Link Register nel seguente modo:
  1. *Salva il contenuto del Link Register nel registro PC*



In Visual

BL	**Indirizzo sottoprogramma**	Salva il PC nel LR prima di effettuare il salto
MOV	**PC, LR**	Rientra da sottoprogramma

Tre registri speciali:
R13 = SP
R14 = LR
R15 = PC

Area di attivazione con annidamento di sottoprogramma

- Nel caso si abbiano diversi sottoprogrammi annidati, prima di chiamare una seconda routine è necessario salvare il **contenuto** del registro LINK_reg (**LR**) per recuperarlo in seguito.
- Si può usare la **pila** per salvare gli indirizzi di rientro delle chiamate annidate all'interno dell'area di attivazione dei programmi chiamanti.
- Il LINK_reg è comunque sempre usato dalle funzioni di Call e Return.
- Alcune architetture non usano il LINK_reg, ma salvano l'indirizzo di rientro solo nella pila. Questa strategia è chiamata **pila a modo implicito**

Esempio

Locazione di memoria		Istruzioni		Commenti
Programma principale				
		⋮		
2000	PRO:	Load	R2, PARAM2	Impila i parametri
2004		Subtract	SP, SP, #4	
2008		Store	R2, (SP)	
2012		Load	R2, PARAM1	
2016		Subtract	SP, SP, #4	
2020		Store	R2, (SP)	
2024		Call	SUB1	Chiama il sottoprogramma
2028		Load	R2, (SP)	Immagazzina il risultato
2032		Store	R2, RIS	
2036		Add	SP, SP, #8	Ripristina il livello della pila
2040		prossima istruzione		
		⋮		

Locazione di memoria		Istruzioni		Commenti
Secondo sottoprogramma				
3000	SUB2:	Subtract	SP, SP, #12	Salva in pila i registri
3004		Store	FP, 8(SP)	
		Store	R2, 4(SP)	
		Store	R3, (SP)	
		Add	FP, SP, #8	Inizializza il puntatore all'area di attivazione
		Load	R2, 4(FP)	Preleva il parametro
		⋮		
		Store	R3, 4(FP)	Impila il risultato di SUB2
		Load	R3, (SP)	Ripristina i registri
		Load	R2, 4(SP)	
		Load	FP, 8(SP)	
		Add	SP, SP, #12	
		Return		Rientro al sottoprogramma 1

Primo sottoprogramma				
2100	SUB1:	Subtract	SP, SP, #24	Salva in pila i registri
2104		Store	LINK_reg,20(SP)	
2108		Store	FP, 16(SP)	
2112		Store	R2, 12(SP)	
2116		Store	R3, 8(SP)	
2120		Store	R4, 4(SP)	
2124		Store	R5, (SP)	
2128		Add	FP, SP, #16	Inizializza il puntatore all'area di attivazione
2132		Load	R2, 8(FP)	Preleva il primo parametro
2136		Load	R3, 12(FP)	Preleva il secondo parametro
		⋮		
		Load	R4, PARAM3	Impila un parametro da passare a SUB2
		Subtract	SP, SP, #4	
		Store	R4, (SP)	
		Call	SUB2	
		Load	R4, (SP)	Spila il risultato ottenuto da SUB2
		Add	SP, SP, #4	
		⋮		
		Store	R5, 8(FP)	Impila la risposta
		Load	R5, (SP)	Ripristina i registri
		Load	R4, 4(SP)	
		Load	R3, 8(SP)	
		Load	R2, 12(SP)	
		Load	FP, 16(SP)	
		Load	LINK_reg,20(SP)	
		Add	SP, SP, #24	
		Return		Rientro al programma principale

Passaggio di Parametri

Passaggio di parametri

- Una routine ha spesso bisogno di:
- Parametri di ingresso su cui operare.
- Restituire un risultato al programma chiamante.

- Esistono due tecniche di passaggio di parametri:

1. Passaggio tramite registri del processore (si usano alcuni registri generici per salvare i parametri). *Numero di parametri limitato dal numero di registri*.
2. Passaggio attraverso la pila (si impilano i parametri nella pila). *Numero di parametri virtualmente illimitato*.

Es1: somma di N numeri tramite registri

Programma chiamante			
	Load	R2, N	Il parametro 1 è la dimensione della lista
	Move	R4, #NUM1	Il parametro 2 è la locazione della lista
	Call	TOTALE	Chiama il sottoprogramma
	Store	R3, SOMMA	Immagazzina il risultato
Sottoprogramma			
TOTALE:	Subtract	SP, SP, #4	Salva in pila R5
	Store	R5, (SP)	
	Clear	R3	Inizializza la somma a 0
CICLO:	Load	R5, (R4)	Preleva il prossimo numero
	Add	R3, R3, R5	Aggiungi questo numero alla somma
	Add	R4, R4, #4	Incrementa di 4 il puntatore
	Subtract	R2, R2, #1	Decrementa il contatore
	Branch_if_[R2]>0	CICLO	
	Load	R5, (SP)	Ripristina il contenuto di R5
	Add	SP, SP, #4	
	Return		Rientra al programma chiamante

Es1: somma di N numeri tramite registri

Programma chiamante			
	Load	R2, N	Il parametro 1 è la dimensione della lista
	Move	R4, #NUM1	Il parametro 2 è la locazione della lista
	Call	TOTALE	Chiama il sottoprogramma
	Store	R3, SOMMA	Immagazzina il risultato
Sottoprogramma			
TOTALE:	Subtract	SP, SP, #4	Salva in pila R5
	Store	R5, (SP)	
	Clear	R3	Inizializza la somma a 0
CICLO:	Load	R5, (R4)	Preleva il prossimo numero
	Add	R3, R3, R5	Aggiungi questo numero alla somma
	Add	R4, R4, #4	Incrementa di 4 il puntatore
	Subtract	R2, R2, #1	Decrementa il contatore
	Branch_if_[R2]>0	CICLO	
	Load	R5, (SP)	Ripristina il contenuto di R5
	Add	SP, SP, #4	
	Return		Rientra al programma chiamante

Es2: somma di N numeri tramite pila

```
Move        R2, #NUM1     Impila i parametri
Subtract    SP, SP, #4
Store       R2, (SP)
Load        R2, N
Subtract    SP, SP, #4
Store       R2, (SP)
Call        TOTALE        Chiama il sottoprogramma (cima della pila a
                          livello 2)
Load        R2, 4(SP)     Spila il risultato e conservalo in SOMMA
Store       R2, SOMMA
Add         SP, SP, #8    Ripristina la cima della pila (cima della pila
                          a livello 1)
⋮
```

```
TOTALE :    Subtract    SP, SP, #16     Salva in pila i registri
            Store       R2, 12(SP)
            Store       R3, 8(SP)
            Store       R4, 4(SP)
            Store       R5, (SP)        (cima della pila a livello 3)
            Load        R2, 16(SP)      Inizializza il contatore a n
            Load        R4, 20(SP)      Inizializza il puntatore alla lista
            Clear       R3              Inizializza la somma a 0
CICLO:      Load        R5, (R4)        Preleva il prossimo numero
            Add         R3, R3, R5      Aggiungi questo numero alla somma
            Add         R4, R4, #4      Incrementa di 4 il puntatore
            Subtract    R2, R2, #1      Decrementa il contatore
            Branch_if_[R2]>0  CICLO
            Store       R3, 20(SP)      Impila il risultato
            Load        R5, (SP)        Ripristina i registri
            Load        R4, 4(SP)
            Load        R3, 8(SP)
            Load        R2, 12(SP)
            Add         SP, SP, #16     (cima della pila a livello 2)
            Return                      Rientra al programma chiamante
```